Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows

Røpke, Stefan; Cordeau, Jean-Francois

Published in:
Transportation Science

Link to article, DOI:
10.1287/trsc.1090.0272

Publication date:
2009

Document Version
Early version, also known as pre-print

Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows

Stefan Ropke
Department of Transport, Technical University of Denmark
Bygningstorvet 115, 2800 Kgs. Lyngby, Denmark
sr@transport.dtu.dk

Jean-François Cordeau
Canada Research Chair in Logistics and Transportation and CIRRELT, HEC Montréal
3000, chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7 Canada
jean-francois.cordeau@hec.ca

October 14, 2008

Abstract

In the pickup and delivery problem with time windows (PDPTW), vehicle routes must be designed to satisfy a set of transportation requests, each involving a pickup and a delivery location, under capacity, time window, and precedence constraints. This paper introduces a new branch-and-cut-and-price algorithm in which lower bounds are computed by solving through column generation the linear programming relaxation of a set partitioning formulation. Two pricing subproblems are considered in the column generation algorithm: an elementary and a non-elementary shortest path problem. Valid inequalities are added dynamically to strengthen the relaxations. Some of the previously proposed inequalities for the PDPTW are also shown to be implied by the set partitioning formulation. Computational experiments indicate that the proposed algorithm outperforms a recent branch-and-cut algorithm.

Keywords: pickup and delivery, column generation, branch-and-price, branch-and-cut, valid inequalities.

1 Introduction

In the *Capacitated Vehicle Routing Problem* (CVRP), a fleet of vehicles based at a common depot must be routed to visit exactly once a set of customers with known demand. Each vehicle route must start and finish at the depot and the total demand of the customers visited by the route must not exceed the vehicle capacity. In the *Vehicle Routing Problem with Time Windows* (VRPTW), a time window is associated with each customer and the vehicle visiting a given customer cannot arrive after the end of the time window. The *Pickup and Delivery Problem with Time Windows* (PDPTW) is a further generalization of the VRPTW in which each customer request is associated with two locations: an origin location where a certain demand must be picked up and a destination where this demand must be delivered. Each route must also satisfy pairing and precedence constraints: for each request, the origin must precede the destination, and both locations must be visited by the same vehicle. The VRPTW can be seen as a special case of the PDPTW in which all requests have a common origin which corresponds to the depot.

The PDPTW has applications in various contexts such as urban courier services, less-than-truckload transportation, and door-to-door transportation services for the elderly and the disabled. In the latter case, narrow time windows are often considered and ride time constraints are imposed to control the time spent by a passenger in the vehicle. The resulting problem is called the *Dial-a-Ride Problem* (DARP).

The CVRP and VRPTW are well known combinatorial optimization problems which have received a lot of attention (see, e.g., Toth and Vigo, 2002). Since it generalizes the VRPTW, the PDPTW is clearly $\mathcal{NP}$-hard. Over the last few decades, several heuristics have been proposed for the PDPTW. However, because of the difficulty of the problem, work on exact methods has been somewhat limited.

Two main approaches have been used to solve the PDPTW exactly: branch-and-price and branch-and-cut. Branch-and-price methods (see, e.g., Barnhart et al., 1998; Desaulniers et al., 1998) use a branch-and-bound scheme in which lower bounds are computed by column generation. The first branch-and-price algorithm for the PDPTW was proposed by Dumas et al. (1991) who considered a set partitioning formulation of the problem in which each column corresponds to a feasible vehicle route and each constraint is associated to a request that must be satisfied exactly once. The resulting pricing subproblem is a shortest path problem with time window, capacity, pairing and precedence constraints. This problem is solvable by dynamic programming and the authors used an algorithm similar to the one developed by Desrosiers et al. (1986) for the single-vehicle pickup and delivery problem with time windows. Several label elimination methods were proposed to accelerate the dynamic programming algorithm, and arc elimination rules were used to reduce the size of the problem. The authors pointed out that their approach works well when the demand of each customer is large with respect to vehicle capacity. The largest instance solved with their approach contains 55 requests.

Another branch-and-price approach for the PDPTW was later described by Savelsbergh and Sol (1998). Their approach differs from that of Desrosiers et al. in several respects: construction and improvement heuristics are used to solve the pricing subproblem; a column management mechanism is used to reduce the size of the column generation master problem; columns are selected with a bias toward increasing the likelihood of identifying feasible

integer solutions; branching is performed on additional variables representing the fraction of a request that is served by a given vehicle; and a primal heuristic is used at each node of the search tree to compute upper bounds. Column generation was also used recently by Xu et al. (2003) and Sigurd et al. (2004) to address variants of the PDPTW arising in long-haul transportation planning and in the transportation of live animals, respectively.

The second family of exact approaches for the PDPTW is branch-and-cut. In branch-and-cut, valid inequalities (i.e., cuts) are added to the formulation at each node of the branch-and-bound tree to strengthen the relaxations which are usually solved by the simplex algorithm. Relying on the previous work of Balas et al. (1995) and Ruland and Rodin (1997) on the *Precedence-Constrained Traveling Salesman Problem* (PCTSP) and the *TSP with Pickup and Delivery* (TSPPD), Cordeau (2006) developed a branch-and-cut algorithm for the DARP based on a three-index formulation of the problem. This algorithm was able to solve instances with four vehicles and 32 requests. It was later improved by Ropke et al. (2007) who compared different formulations of the DARP and PDPTW, and introduced two new families of inequalities for these problems. One is an adaptation of the *reachability cuts* introduced by Lysgaard (2006) for the VRPTW, while the other is called *fork inequalities*. Both families can also be used in the context of column generation and will be described in Section 4. Using these inequalities, Ropke et al. were able to solve DARP instances with eight vehicles and 96 requests. Another branch-and-cut approach, based on a two-index formulation, was proposed by Lu and Dessouky (2004). This formulation contains a polynomial number of constraints, but relies on extra variables to impose pairing and precedence constraints. Instances with up to five vehicles and 25 requests were solved optimally with this approach.

For reviews on pickup and delivery problems, the reader is referred to the works of Savelsbergh and Sol (1995), Desaulniers et al. (2002) and Cordeau et al. (2007). A polyhedral study of the TSPPD was also recently performed by Dumitrescu et al. (2008).

In this paper, we introduce a new branch-and-cut-and-price algorithm for the PDPTW. It is well known that set partitioning formulations of vehicle routing problems tend to provide stronger lower bounds than formulations based on arc (flow) variables (see Bramel and Simchi-Levi, 2002). Another motivation for studying branch-and-cut-and-price algorithms for the PDPTW is that the most successful exact algorithms for the related CVRP and VRPTW are of this type. For the CVRP the best results have been obtained by Fukasawa et al. (2006) and Baldacci et al. (2008), while the best results for the VRPTW have been presented by Jepsen et al. (2008) and Desaulniers et al. (2008). A branch-and-price approach for a related problem involving simultaneous pickups and deliveries was also described by Dell'Amico et al. (2006).

Two different shortest path problems have been considered as pricing subproblems for the PDPTW in the literature. In the first application of column generation to the PDPTW (Dumas et al., 1991) a non-elementary shortest path problem was solved, while later ones (Sol, 1994; Sigurd et al., 2004) have used an elementary shortest path problem. Both of these problems are $\mathcal{NP}$-hard. Little is known about how the relaxations obtained by solving these two subproblems differ. The only result we are aware of is by Sol (1994) who has shown an example where the objective of the relaxation obtained with the non-elementary problem is half of the objective obtained with the elementary one.

The contributions of this paper are threefold. First, we show that introducing valid inequalities in the set partitioning formulation, expressed in the variables of the compact

formulation, ruins an important property of the cost matrix of the pricing problem. We show how the property can be reestablished by perturbing the cost matrix of the pricing problem. This allows us to use a fast algorithm for the pricing problem together with valid inequalities in the set partitioning formulation. Second, we show that some previously proposed inequalities are implied by the LP relaxation of the set partitioning formulation and that the elementary relaxation implies more inequalities than the non-elementary one. Third, we report extensive computational experiments on a large set of instances.

The remainder of the paper is organized as follows. Section 2 defines the PDPTW and introduces mathematical formulations of the problem. Section 3 discusses the two pricing subproblems that we use within the branch-and-price algorithm. Section 4 describes valid inequalities that can be added to the formulation, while Section 5 studies the relationships between these inequalities and the set partitioning formulation of the problem. The branch-and-cut-and-price algorithm is then described in Section 6. Finally, computational results are reported in Section 7, followed by conclusions in the last section.

2 Mathematical Formulation

In this section, we introduce the notation that is used throughout the paper. We then present a standard three-index model of the problem, followed by a set partitioning formulation.

2.1 Notation

Let n denote the number of requests to satisfy. We define the PDPTW on a directed graph $G = (N, A)$ with node set $N = \{0, \ldots, 2n+1\}$ and arc set A. Nodes 0 and $2n+1$ represent the origin and destination depots, while subsets $P = \{1, \ldots, n\}$ and $D = \{n+1, \ldots, 2n\}$ represent pickup and delivery nodes, respectively. Thus, each request i is associated with a pickup node i and a delivery node $n+i$.

With each node $i \in N$ are associated a load q_i and a non-negative service duration d_i satisfying $q_0 = q_{2n+1} = 0$, $q_i = -q_{n+i}$ $(i = 1, \ldots, n)$ and $d_0 = d_{2n+1} = 0$. A time window $[a_i, b_i]$ is also associated with every node $i \in P \cup D$, where a_i and b_i represent the earliest and latest time, respectively, at which service may start at node i. The depot nodes may also have time windows $[a_0, b_0]$ and $[a_{2n+1}, b_{2n+1}]$ representing the earliest and latest times, respectively, at which the vehicles may leave from and return to the depot. Let K denote the set of vehicles. We assume that vehicles are identical and have capacity Q. With each arc $(i, j) \in A$ are associated a routing cost c_{ij} and a travel time t_{ij}. In the remainder of the paper, we assume that the travel time t_{ij} includes the service time d_i at node i. We also assume that the triangle inequality holds both for routing costs and travel times.

2.2 Three-index formulation of the PDPTW

For each arc $(i, j) \in A$ and each vehicle $k \in K$, let x_{ij}^k be a binary variable equal to 1 if and only if vehicle k travels directly from node i to node j. For each node $i \in N$ and each vehicle $k \in K$, let B_i^k be the time at which vehicle k begins service at node i, and Q_i^k be the load of vehicle k upon leaving node i. Using these variables, the PDPTW can be formulated as

the following non-linear mixed-integer program (see, e.g., Cordeau (2006)):

$$\text{Min} \quad \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ij}^k \tag{1}$$

subject to

$$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1 \qquad \forall i \in P \tag{2}$$

$$\sum_{j\in N} x_{ij}^k - \sum_{j\in N} x_{n+i,j}^k = 0 \qquad \forall i \in P, k \in K \tag{3}$$

$$\sum_{j\in N} x_{0j}^k = 1 \qquad \forall k \in K \tag{4}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{j\in N} x_{ij}^k = 0 \qquad \forall i \in P \cup D, k \in K \tag{5}$$

$$\sum_{i\in N} x_{i,2n+1}^k = 1 \qquad \forall k \in K \tag{6}$$

$$B_j^k \geq (B_i^k + t_{ij})x_{ij}^k \qquad \forall i \in N, j \in N, k \in K \tag{7}$$

$$Q_j^k \geq (Q_i^k + q_j)x_{ij}^k \qquad \forall i \in N, j \in N, k \in K \tag{8}$$

$$B_i^k + t_{i,n+i} \leq B_{n+i}^k \qquad \forall i \in P, k \in K \tag{9}$$

$$a_i \leq B_i^k \leq b_i \qquad \forall i \in N, k \in K \tag{10}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{Q, Q + q_i\} \qquad \forall i \in N, k \in K \tag{11}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall i \in N, j \in N, k \in K. \tag{12}$$

The objective function (1) minimizes the total routing cost. Constraints (2) and (3) ensure that each request is served exactly once and that the pickup and delivery nodes are visited by the same vehicle. Constraints (4)-(6) guarantee that the route of each vehicle k starts at the origin depot and ends at the destination depot. Consistency of the time and load variables is ensured by constraints (7) and (8). Constraints (9) ensure that for each request i, the pickup node is visited before the delivery node. Finally, inequalities (10) and (11) impose time windows and capacity constraints, respectively. The model is non-linear because of inequalities (7) and (8) but it can easily be linearized by using standard reformulation techniques.

2.3 Set partitioning formulation of the PDPTW

To formulate the problem as a set partitioning problem, let Ω denote the set of all feasible routes satisfying constraints (3)-(12), dropping index k (as all vehicles are assumed to be identical). For each route $r \in \Omega$, let c_r be the cost of the route and let a_{ir} be a constant indicating the number of times node $i \in P$ is visited by r. Let also y_r be a binary variable

equal to 1 if and only if route $r \in \Omega$ is used in the solution. The PDPTW can then be formulated as the following set partitioning problem:

$$\text{Min} \quad \sum_{r \in \Omega} c_r y_r \tag{13}$$

subject to

$$\sum_{r \in \Omega} a_{ir} y_r = 1 \qquad \forall i \in P \tag{14}$$

$$y_r \in \{0, 1\} \qquad \forall r \in \Omega. \tag{15}$$

The objective function (13) minimizes the cost of the chosen routes while constraints (14) ensure that every request is served once. A lower bound on the optimal value of (13)-(15) can be obtained by solving the linear programming (LP) relaxation which is obtained by replacing the integrality requirements (15) with the constraints

$$y_r \geq 0 \qquad \forall r \in \Omega. \tag{16}$$

Because of the large size of set Ω, it is usually very difficult to solve or even to represent model (13)-(15) explicitly. Instead, its LP relaxation is solved using column generation. In a column generation approach, a restricted master problem is obtained by considering a subset $\bar{\Omega} \subseteq \Omega$ of routes. Additional columns of negative reduced cost are generated by solving a *pricing subproblem*. Following Wolsey (1998), we call the problem defined by (13)–(15) the *integer programming master problem* (*IPM*) and its LP relaxation the *linear programming master problem* (*LPM*). The pricing problem for the PDPTW is

$$\text{Min} \quad \sum_{i,j \in N} d_{ij} x_{ij} \tag{17}$$

subject to constraints (3)–(12) (dropping index k), where d_{ij} is defined as

$$d_{ij} = \begin{cases} c_{ij} - \pi_i & \forall i \in P, j \in N \\ c_{ij} & \forall i \in N \setminus P, j \in N, \end{cases} \tag{18}$$

and π_i is the dual variable associated with the set partitioning constraint (14) for node i. We denote this problem as SP1.

The definition of d_{ij} in equation (18) ensures that $d_{ij} + d_{jk} \geq d_{ik}$ if j is a delivery node as c_{ij} satisfies the triangle inequality. We say that a cost matrix that satisfies this property satisfies the *delivery triangle inequality*. As will be shown in Section 3 this is computationally convenient. The problem defined by objective (17) and constraints (3)–(12) is a constrained shortest path problem called the *Elementary Shortest Path Problem with Time Windows, Capacity, and Pickup and Delivery* (ESPPTWCPD). In Section 3 we explain how this and related problems can be solved using label setting algorithms.

Instead of solving the shortest path problem SP1 one can solve relaxed versions of this problem. A relaxed shortest path problem implies that a set of routes Ω' is implicitly considered, where $\Omega \subseteq \Omega'$. If Ω' satisfies the property that none of the columns from the set

$\Omega' \setminus \Omega$ can be used in a feasible integer solution to IPM, then the set partitioning problem solved on Ω' will have the same set of optimal solutions as the one solved on Ω. Obviously, the lower bound obtained by solving the LP relaxation on Ω' may, however, be weaker. An example of a relaxation of the elementary shortest path problem that satisfies this property consists of allowing cycles in the path. In this case, some requests may be served more than once. Paths containing cycles cannot, however, appear in a feasible integer solution because of constraints (14). This relaxation was used by Dumas et al. (1991) and it is described in more detail in Section 3.2.

Relaxations inducing sets Ω' for which one cannot ensure that no column from $\Omega' \setminus \Omega$ can belong to a feasible integer solution to IPM can also be used. In this case, however, valid inequalities must be added to the master problem to render such solutions infeasible. This approach was used by Ropke (2005) to solve the PDPTW using more relaxed pricing problems.

Valid inequalities expressed in terms of the x_{ij}^k variables from the three-index formulation (1)-(12) can be added to the master problem following the approach proposed by Kohl et al. (1999) for the VRPTW. Since all vehicles are identical, we first observe that such inequalities can be expressed in terms of x_{ij} variables and can be written in the form

$$\sum_{i=0}^{2n+1} \sum_{j=0}^{2n+1} \alpha_{ij} x_{ij} \geq \beta,$$

where $\alpha_{ij} \in \mathbb{R}$ is the coefficient of arc $(i, j) \in A$ and $\beta \in \mathbb{R}$ is a constant. This inequality is transfered to the master problem as

$$\sum_{r \in \Omega} \phi_r y_r \geq \beta,$$

where $\phi_r = \sum_{(i,j) \in A} \psi_{ijr} \alpha_{ij}$, and ψ_{ijr} is the number of times arc (i, j) is used in route r. The introduction of a valid inequality in the master problem modifies the pricing problem. Indeed, the arc costs d_{ij} are now defined as follows:

$$d_{ij} = \begin{cases} c_{ij} - \pi_i - \alpha_{ij}\mu & \forall i \in P, j \in N \\ c_{ij} - \alpha_{ij}\mu & \forall i \in N \setminus P, j \in N, \end{cases} \tag{19}$$

where μ is the dual variable associated with the added inequality. Any number of inequalities can be added in this way. Notice that unlike definition (18) this new definition of d_{ij} does not satisfy the delivery triangle inequality.

3 Constrained Shortest Path Problems

Resource constrained shortest path problems arising in column generation approaches for vehicle routing problems are typically solved using dynamic programming techniques called labeling algorithms. One seeks a shortest (cheapest) path from a source node s to a sink node t in the graph $G = (N, A)$. Each arc in the graph has an associated cost and the cost of a path is simply the sum of the costs of the arcs used in the path. The paths must

satisfy conditions on resources usage between the source and sink. Time is an example of a resource that is consumed along the path, and time windows on the individual nodes are an example of conditions that must be satisfied. An overview of resource constrained shortest path problems and of appropriate solution techniques was given by Irnich and Desaulniers (2005).

Labeling algorithms build partial paths in the graph G. Each partial path starts at s and ends at a node $i \in N$. Existing partial paths are extended along the arcs leaving the end node of the partial path and the algorithm in principle constructs all feasible paths in the graph by starting with the partial path containing only node s. In order to speed up the algorithm, *dominated* paths are removed during the search. A path p dominates another path p' if they both end at the same node $i \in N$, the cost of p is less than or equal to the cost of p', and all feasible extensions of p' by a partial path to the sink node are also feasible for p. As it can be difficult to check if a path dominates another path using this definition, one may introduce *sufficient* (but not *necessary*) conditions for dominance. Examples of such conditions are provided in Sections 3.1 and 3.2 below. Partial paths are represented by so-called *labels* that record the resource consumption at the end node of the partial path and dominance criteria are defined in terms of the labels (see, e.g., Irnich and Desaulniers (2005)).

3.1 ESPPTWCPD - SP1

The ESPPTWCPD, denoted by SP1, is the natural pricing problem for the PDPTW. In the context of the PDPTW, it was first used by Sol (1994) and later by Sigurd et al. (2004) for a PDPTW with additional precedence constraints. Sigurd et al. (2004) have described a general labeling algorithm for the ESPPTWCPD and a more efficient one that takes advantage of the additional precedence constraints. In this section we introduce a new labeling algorithm for the ESPPTWCPD that contains a less restrictive, sufficient dominance condition with respect to the algorithm proposed by Sol (1994) and the general one described by Sigurd et al. (2004). A less restrictive dominance condition implies that more labels can be eliminated, resulting in a better performance of the shortest path algorithm. Our label setting algorithm is described in detail in Ropke and Cordeau (2008), and in this section we present the main ideas. In what follows we assume that the source and sink nodes are, respectively, 0 and $2n + 1$.

For each label we store the following data: η – the node of the label, t – the arrival time at the node, l – the load of the vehicle after visiting node η, c – the accumulated cost, $\mathcal{O} \subseteq \{1, \ldots, n\}$ – the set of requests that have been started but not completed, i.e., the pickup has been served but not the delivery, $U \subseteq \{1, \ldots, n\}$ – the set of *unreachable* requests. A request i is said to be unreachable if its pickup node has already been visited on the partial path or if going straight from η to pickup node i would violate the time window at node i. We do not consider the delivery node of request i in the definition of unreachable requests because of the preprocessing steps mentioned in Section 6. The preprocessing ensures that $b_i + t_{i,n+i} \leq b_{n+i}, \forall i \in P$. We also store a pointer to the parent label in each label. Our resources are t, l, c, U and $\mathcal{O}$. The notation $t(L)$ is used to refer to the arrival time in label L and similar notation is used for the rest of the resources (e.g., $\eta(L)$, $l(L)$, $c(L)$, $U(L)$ and $\mathcal{O}(L)$). The requests in $\mathcal{O}$ are said to be *open*

When extending a label L along an arc $(\eta(L), j)$, the extension is legal only if $t(L) + t_{\eta(L),j} \leq b_j$ and $l(L) + q_j \leq Q$, ensuring that time window and vehicle capacity are obeyed. Furthermore, L and j must satisfy one of the following three conditions which ensure that the extension is compatible with the sets $u(L)$ and $\mathcal{O}(L)$:

$$0 < j \leq n \quad \wedge \quad j \notin u(L) \tag{20}$$

$$n < j \leq 2n \quad \wedge \quad j - n \in \mathcal{O}(L) \tag{21}$$

$$j = 2n + 1 \quad \wedge \quad \mathcal{O}(L) = \emptyset. \tag{22}$$

The dominance condition used in the label-setting algorithm is the following: a label L_1 dominates a label L_2 if

$$\eta(L_1) = \eta(L_2),\ t(L_1) \leq t(L_2),\ c(L_1) \leq c(L_2),\ U(L_1) \subseteq U(L_2),\ \mathcal{O}(L_1) \subseteq \mathcal{O}(L_2). \tag{23}$$

The validity of the dominance condition and its relation to that proposed by Sol (1994) is discussed in Ropke and Cordeau (2008). The dominance condition is constructed by combining ideas from Dumas et al. (1991) and Feillet et al. (2004). It is important to point out that the dominance condition is valid only if the delivery triangle inequality holds. When this assumption holds we are sure that visiting a delivery node never makes the path cheaper. Without the delivery triangle inequality the last condition in (23) would be $\mathcal{O}(L_1) = \mathcal{O}(L_2)$ which is more restrictive and therefore not as strong as the condition in (23). The definition of arc costs d_{ij} from equation (19) does not satisfy the delivery triangle inequality, but in Section 3.3 we show that this is not a major problem as any cost matrix can be transformed into one that satisfies the delivery triangle inequality while maintaining the cost of every feasible path.

Another limitation of the dominance criteria is that the removal of arcs from the network must be performed carefully. An arc (i, j) cannot be removed if the subpath (i, k, j) is valid for some delivery node k (see Ropke and Cordeau (2008) for details). Arc elimination is useful within a branch-and-bound scheme that branches on the arcs in the original formulation (1)-(12). This limitation is one of the reasons for the choice of branching rules in Section 6.3.

3.2 SPPTWCPD - SP2

We now consider the *Shortest Path Problem with Time Windows, Capacity, and Pickup and Delivery* (SPPTWCPD), denoted SP2, which relaxes SP1 by not requiring paths to be elementary. In this problem we do, however, impose two conditions which help prevent cycles: *i)* after performing a pickup, the same pickup cannot be performed again before the corresponding delivery has been performed, and *ii)* a delivery cannot be performed before the corresponding pickup has been performed. These conditions ensure that any cycle in a path will contain at least four nodes. The shortest cycle is of the form $i \rightarrow n + i \rightarrow j \rightarrow i$. One cannot go from $n + i$ to i as the corresponding arc does not exist in our graph. If time windows are tight, such cycles are unlikely to arise and the SPPTWCPD should yield good lower bounds. This shortest path problem was used as a pricing problem by Dumas et al. (1991).

In the label-setting algorithm for the SPPTWCPD we store η, t, l, c and $\mathcal{O}$ together with a pointer to the parent label, as explained in Section 3.1, but we do not store the set

U. Determining whether an extension of a label is feasible is done in a similar way as for SP1 but condition (20) is replaced with

$$0 < j \leq n \quad \wedge \quad j \notin \mathcal{O}(L). \tag{24}$$

Replacing condition (20) with (24) implies that non elementary paths can be generated. When the delivery of request i has been performed, i is removed from $\mathcal{O}$ and the path may again visit the pickup node of request i.

The dominance criterion employed, is the following: a label L_1 dominates a label L_2 if

$$\eta(L_1) = \eta(L_2), \quad t(L_1) \leq t(L_2), \quad c(L_1) \leq c(L_2), \quad \mathcal{O}(L_1) \subseteq \mathcal{O}(L_2).$$

Dumas et al. (1991) showed that this dominance criterion is valid. The concerns about the distance matrix and missing arcs that were discussed in Section 3.1 apply to the dominance condition for the SPPTWCPD as well.

3.2.1 Label elimination

Dumas et al. (1991) have proposed rules for eliminating labels that cannot be extended to node $2n+1$. The key observation is that given a label L one can examine the deliveries of the open requests in $\mathcal{O}(L)$. If it is impossible to create a path from $\eta(L)$ to node $2n+1$ that visits exactly all of the deliveries of $\mathcal{O}(L)$ and that satisfies all time windows, then label L can be discarded because of the triangle inequality on t_{ij}. Without the triangle inequality assumption the argument is invalid because detours may decrease the travel time. Determining whether such a path exists can be done by solving a *Traveling Salesman Problem with Time Windows* (TSPTW), which is $\mathcal{NP}$-hard. Consequently, Dumas et al. (1991) have proposed to consider only subsets of $\mathcal{O}(L)$ of cardinality one and two. We use the same approach here. In addition, we also test two subsets containing three deliveries. In the first subset, the first delivery, i_1, is the one farthest from $\eta(L)$, the next delivery, i_2, is the one farthest from $\eta(L)$ and i_1, and the last delivery is the one farthest from $\eta(L)$, i_1 and i_2. In the second subset we choose the deliveries that have the earliest deadline b_i. When the subset has been chosen we solve the resulting TSPTW by simple enumeration. The label elimination works for both pricing problems considered in this paper.

3.3 Transforming the pricing problem cost matrix

In Sections 3.1 and 3.2 it was shown how effective dominance criteria could be devised for SP1 and SP2 when the cost matrix for the pricing problem satisfies the *delivery triangle inequality*. In Section 2.3 we saw that the pricing problem cost matrix does not satisfy the property when cuts have been added to the master problem. In this section we explain how it is possible to transform an arbitrary cost matrix into a cost matrix that satisfies the delivery triangle inequality while maintaining the optimal solutions of SP1 and SP2.

Lemma 1. For any vector $(\theta_1, \ldots, \theta_{|P|}) \in \mathbb{Q}^{|P|}$, let the cost matrix $(\tilde{d}_{ij})$ be defined by

$$\begin{array}{lllllll} \tilde{d}_{ij} & = & d_{ij} - \theta_i, & \tilde{d}_{n+i,j} & = & d_{n+i,j} + \theta_i & \forall i \in P, \forall j \in N \\ \tilde{d}_{0j} & = & d_{0j} & \tilde{d}_{2n+1,j} & = & d_{2n+1,j} & \forall j \in N. \end{array}$$

Using $(\tilde{d}_{ij})$ instead of (d_{ij}) does not change the cost of any feasible path in SP1 or SP2.

Proof. As any feasible path visiting node $i \in P$ also has to visit node $n+i \in D$ and vice-versa the sum of the θ_i terms sums to zero in any feasible path. □

Note that although the definition of $\tilde{d}_{2n+1,j}$ is included in Lemma 1 to define the entire matrix $(\tilde{d}_{ij})$, it is never used in practice as no arc leaves node $2n+1$. The next proposition shows how to select the modifiers θ_i such that $\tilde{d}_{ij}$ satisfies the delivery triangle inequality.

Proposition 1. If

$$\theta_j \geq d_{ik} - (d_{i,n+j} + d_{n+j,k}) \quad \forall j \in P, \forall i,k \in N$$

then $(\tilde{d}_{ij})$ defined in Lemma 1 satisfies

$$\tilde{d}_{ij} + \tilde{d}_{jk} \geq \tilde{d}_{ik} \quad \forall i,k \in N, \forall j \in D.$$

Proof. By distinguishing between the four cases $i = 0$, $i \in P$, $i \in D$, $i = 2n+1$ and substituting the definition of $\tilde{d}_{ij}$ into the expression $\tilde{d}_{ij} + \tilde{d}_{jk}$ we obtain the desired result. For example for $i \in P$ we obtain

$$\begin{aligned}
\tilde{d}_{ij} + \tilde{d}_{jk} &= d_{ij} - \theta_i + d_{jk} + \theta_{j-n} \\
&\geq d_{ij} - \theta_i + d_{jk} + \max_{i',k' \in N}\{d_{i'k'} - (d_{i'j} + d_{jk'})\} \\
&\geq d_{ij} - \theta_i + d_{jk} + (d_{ik} - (d_{ij} + d_{jk})) \\
&= d_{ik} - \theta_i \\
&= \tilde{d}_{ik}
\end{aligned}$$

for all $j \in D$ and $k \in N$. □

In practice one can use the modifiers $\theta_j = \max_{i,k \in N}\{d_{ik} - (d_{i,n+j} + d_{n+j,k})\}, \forall j \in P$ which can be calculated in $O(n^3)$ time. Note that the transformation ensures that $\tilde{d}_{ij}$ satisfies the delivery triangle inequality even if the original cost matrix (c_{ij}) does not.

3.4 Possible improvements

Irnich and Villeneuve (2006) have proposed a labeling algorithm that solves non-elementary shortest path problems while ensuring that cycles of length k or smaller do not occur. Their approach could be used to strengthen the lower bound of the LPM when using SP2 as a pricing problem since SP2 allows cycles containing more than two arcs. However, the computational results presented in Section 7 show that the lower bounds obtained with SP2 are already quite close to the lower bounds obtained with SP1, so it is not clear whether the extra effort to forbid longer cycles is worthwhile.

On a different note, Righini and Salani (2006) have proposed a bi-directional approach to shortest path problems with resource constraints. Instead of generating partial paths only from the source node, they simultaneously extend paths both from the source and the sink nodes. The two searches eventually meet at a point where the paths from the source are merged with paths from the sink. This approach has shown great potential for reducing the running time of the shortest path algorithms used in the branch-and-price methods for the CVRP and VRPTW. However, this approach does not seem promising

for the PDPTW: the forward path extension can be performed as described above but the backward extension would need a more restrictive dominance condition. The point is that for the strong dominance criterion to work for the backward extension the matrix (d_{ij}) should satisfy $d_{ij} + d_{jk} \geq d_{ik}$, $\forall j \in P, i, k \in N$ and we have not found a way to ensure this while maintaining the delivery triangle inequality property of (d_{ij}). A prototype implementation of a bidirectional search that used a more restrictive dominance condition criterion for the backward extension gave disappointing results.

Another interesting approach for improving label setting algorithms used for solving resource constrained elementary shortest path problems was proposed by Boland et al. (2006) and by Righini and Salani (2008) and used in the most successful exact algorithm for the VRPTW to date (Desaulniers et al. (2008)). The basic idea of this approach is to iteratively solve a series of relaxed shortest paths problems. In the first problem all nodes can be visited multiple times. The solution to this problem is inspected and a set of nodes S is selected. In subsequent problems, the nodes in S can be visited at most once. After each iteration the set S is enlarged until the optimal path for the relaxed problem is elementary. The motivation for the algorithm is that one does not have to include all nodes in S before reaching an elementary path and that shortest path computations with a limited set S can be much faster compared to requiring that all nodes should be visited at most once. This approach can also be applied to the ESPPTWCPD by limiting the nodes that can be included in the set U to just a subset of all requests. What is perhaps more interesting is that the approach also could be applied to the SPPTWCPD by first solving a pure SPPTWC and gradually enforcing that request pairs should satisfy pairing and precedence constraints. An advanced algorithm for the ESPPTWCPD would also start out from the pure SPPTWC and gradually enforce both elementarity, pairing and precedence constraints. We have not performed experiments with such algorithms.

4 Valid Inequalities

In this section we describe several families of valid inequalities that have been used in the branch-and-cut algorithms proposed by Cordeau (2006) and by Ropke et al. (2007). For two of these families, rounded capacity inequalities and precedence inequalities, we provide strengthened inequalities. We also show how the so-called 2-path cuts for the VRPTW can be applied to the PDPTW. All of these inequalities are useful in strengthening the LP-relaxation of two-index and three-index formulations of the PDPTW, but as will be shown in Section 5 some of them are in fact implied by the LP relaxation of the set-partitioning formulations considered in this paper.

To describe these inequalities, it is convenient to introduce new notation. For any node subset $S \subseteq N$, let $\delta^+(S) = \{(i,j) \in A | i \in S, j \notin S\}$. We also use the notation $\delta^+(i)$ to designate the set $\delta^+(\{i\})$. Also let $\pi(S) = \{i \in P | n+i \in S\}$ and $\sigma(S) = \{n+i \in D | i \in S\}$ denote the sets of *predecessors* and *successors* of S. Finally, let $x_{ij} = \sum_{k \in K} x_{ij}^k$.

4.1 Infeasible path inequalities

Cordeau (2006) and Ropke et al. (2007) have discussed infeasible path inequalities and

various strengthenings for the PDPTW. In this paper we use two types of infeasible path inequalities. Consider an infeasible path $R = (k_1, \ldots, k_\rho)$, then the inequality

$$\sum_{i=1}^{\rho-1} x_{k_i,k_{i+1}} \leq \rho - 2 \tag{25}$$

is valid.

If $k_1 = i$ and $k_\rho = n+i$ for some $i \in P$ and the path is infeasible because of time windows then Cordeau (2006) has observed that the inequality can be strengthened to

$$\sum_{i=1}^{\rho-1} x_{k_i,k_{i+1}} \leq \rho - 3. \tag{26}$$

4.2 Fork inequalities

Let $R = (k_1, \ldots, k_\rho)$ be a path in G and $S, T_1, \ldots, T_\rho \subset (P \cup D)$ be subsets such that $k_j \notin T_{j-1}$ for $j = 2, \ldots, \rho$. If for any integer $h \leq \rho$ and any node pair $i \in S, j \in T_h$, the path $(i, k_1, \ldots, k_h, j)$ is infeasible, then the following inequality is valid for the PDPTW:

$$\sum_{i \in S} x_{i,k_1} + \sum_{h=1}^{\rho-1} x_{k_h,k_{h+1}} + \sum_{h=1}^{\rho} \sum_{j \in T_h} x_{k_h,j} \leq \rho. \tag{27}$$

Similarly, let $R = (k_1, \ldots, k_\rho)$ be a path in G and $S_1, \ldots, S_\rho, T \subset (P \cup D)$ be subsets such that $k_j \notin S_{j+1}$ for $j = 1, \ldots, \rho - 1$. If for any integer $h \leq \rho$ and any node pair $i \in S_h, j \in T$, the path $(i, k_h, \ldots, k_\rho, j)$ is infeasible, then the following inequality is valid for the PDPTW:

$$\sum_{h=1}^{\rho} \sum_{i \in S_h} x_{i,k_h} + \sum_{h=1}^{\rho-1} x_{k_h,k_{h+1}} + \sum_{j \in T} x_{k_\rho,j} \leq \rho. \tag{28}$$

Inequalities (27) and (28) were introduced by Ropke et al. (2007) and are called *outfork* and *infork inequalities*, respectively. We refer to the paper by Ropke et al. (2007) for examples and figures.

4.3 Rounded capacity inequalities

Rounded capacity inequalities which are often used in the context of the CVRP (see, e.g., Naddef and Rinaldi, 2002) can also be used for the PDPTW. For any node subset $S \subseteq P \cup D$, let $\kappa(S)$ be a lower bound on the number of times that vehicles must enter the set. The following inequality is then clearly valid:

$$x(\delta^+(S)) \geq \kappa(S). \tag{29}$$

Cordeau (2006) and Ropke et al. (2007) have proposed to use $\kappa(S) = \max\left\{1, \left\lceil \frac{|q(S)|}{Q} \right\rceil \right\}$. This lower bound can be improved to $\kappa(S) = \max\left\{1, \left\lceil \frac{q(\pi(S)\backslash S)}{Q} \right\rceil, \left\lceil \frac{-q(\sigma(S)\backslash S)}{Q} \right\rceil \right\}$. The bound is

valid as $q(\pi(S)\backslash S)$ is a lower bound on the load of the vehicles entering set S and $-q(\sigma(S)\backslash S)$ is a lower bound on the load of the vehicles leaving S. This new lower bound is stronger than the previous one because

$$\begin{aligned} |q(S)| &= |q(S \cap D) + q(S \cap P)| \\ &= |q(\pi(S)) + q(\sigma(S))| \\ &= |q(\pi(S) \setminus S) + q(\sigma(S) \setminus S)| \\ &\leq \max\{q(\pi(S) \setminus S), -q(\sigma(S) \setminus S)\}. \end{aligned}$$

Furthermore, examples can be constructed where the inequality is strict. The second equality holds because $q(S \cap D) = -q(\pi(S))$ and $q(S \cap P) = -q(\sigma(S))$, the third equality holds because $q(\{i, n+i\}) = 0$, and the inequality holds because $q(\pi(S)\backslash S) \geq 0$ and $q(\sigma(S)\backslash S) \leq 0$.

4.4 2-path inequalities

2-path inequalities are a variation of the rounded capacity inequalities and were proposed for the VRPTW by Kohl et al. (1999). When the set $S \subseteq P \cup D$ cannot be served by a single path, the following inequality is valid:

$$x(\delta^+(S)) \geq 2. \tag{30}$$

To determine whether the set S can be served by a single path one can go further than just considering capacities as in Section 4.3. Indeed, for the VRPTW Kohl et al. (1999) proposed to solve a TSPTW on the set of nodes: if the TSPTW is infeasible then at least two paths are needed to visit all nodes in S.

For the PDPTW a similar approach can be used. If it is impossible to find a tour serving all nodes in S while satisfying precedence, capacity and time window constraints then any feasible solution must use at least two arcs from the set $\delta^+(S)$. The idea can be taken further by observing that if a path serves all nodes in S by entering and leaving the set once, then the nodes $\pi(S) \setminus S$ must be served by this path before entering S and the nodes $\sigma(S) \setminus S$ must be served after leaving S. If such a path cannot be found then S defines a valid inequality (30) even though there exists a tour through S satisfying precedence, capacity and time window constraints.

4.5 Reachability inequalities

For any node $i \in N$, let $A_i^- \subset A$ be the minimum arc set such that any feasible path from the origin depot 0 to node i uses only arcs from A_i^-. Let also A_i^+ be the minimum arc set such that any feasible path from i to the destination depot $2n+1$ uses only arcs in A_i^+. Consider a node set T such that each node in T must be visited by a different vehicle. This set is said to be *conflicting*. For any conflicting node set T, define the *reaching arc set* $A_T^- = \cup_{i \in T} A_i^-$ and the *reachable arc set* $A_T^+ = \cup_{i \in T} A_i^+$. For any node set $S \subseteq P \cup D$ and any conflicting node set $T \subseteq S$, the following two valid inequalities were introduced by Lysgaard (2006) for the VRPTW:

$$x(\delta^-(S) \cap A_T^-) \geq |T| \tag{31}$$

$$x(\delta^+(S) \cap A_T^+) \geq |T|. \tag{32}$$

These inequalities are obviously also valid for the PDPTW. In this problem, however, nodes can be conflicting not only because of time windows but also because of the interactions with the precedence relationships and the capacity constraints.

4.6 Precedence inequalities

Let $S \subset N$ be a subset such that $i, 2n+1 \in S$ and $0, n+i \notin S$ for some $i \in P$. Then the following inequality is valid:

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq 1.$$

Precedence inequalities were introduced by Ruland and Rodin (1997) in the context of the TSP with pickup and delivery.

4.7 Strengthened precedence inequalities

Using ideas from the reachability inequalities, precedence inequalities can be strengthened as follows.

Proposition 2. Let A_i be the set of arcs that can be used in a feasible path from i to $n+i$. Furthermore let $S \subset N$ be a subset such that $i, 2n+1 \in S$ and $0, n+i \notin S$ for some $i \in P$. Then the following inequality is valid for the PDPTW:

$$\sum_{(i,j)\in(\delta^+(S)\cap A_i)} x_{ij} \geq 1. \tag{33}$$

Proof. Assume that inequality (33) is violated in a feasible integer solution. This implies that there exists a request $i \in P$ and a set $S \subset N$ such that $i, 2n+1 \in S$, $0, n+i \notin S$ and $\sum_{(i,j)\in(\delta^+(S)\cap A_i)} x_{ij} = 0$. Let p be the path in the solution that visits node i. To be part of a feasible solution, path p must visit node $n+i$ after visiting node i. Let p' be the subpath of p that starts in i and ends in $n+i$. It is clear that p' must use at least one arc from $\delta^+(S)$ because $i \in S$ and $n+i \notin S$. From the definition of A_i it is also clear that the arcs in p' all belong to the set A_i. Consequently, $\sum_{(i,j)\in(\delta^+(S)\cap A_i)} x_{ij} \geq 1$. □

5 Relationship Between Set Partitioning Formulations and Valid Inequalities

This section explores the relationship between the set partitioning formulations using SP1 and SP2 as pricing problems and the valid inequalities presented in Section 4. It turns out that several families of valid inequalities are implied by the set partitioning formulation with the SP1 subproblem. No such analysis was previously performed for the PDPTW, but some results were obtained by Letchford and Salazar González (2006) for set partitioning relaxations of the CVRP and VRPTW. Such a study is not only interesting from a theoretical point of view, but it is also useful from a practical perspective as it implies that we do not need to consider separation procedures for certain classes of valid inequalities.

Given a solution y^* to the LP relaxation of the set partitioning formulation given by (13), (14) and (16) one can obtain the corresponding two-index solution x^* given by $x^*_{ij} = \sum_{r\in\Omega} y^*_r \psi_{ijr}, \forall i, j \in N$, where ψ_{ijr} is a constant indicating the number of times arc (i, j) is used in route r. Let Ω^*_i be the set of columns serving request i in the current solution, i.e., $\Omega^*_i = \{r \in \Omega : y^*_r > 0, a_{ir} \geq 1\}$.

We first show that the fork inequalities are implied by SP1.

Lemma 2. Let p be a path generated by SP1 that visits $q \leq \rho$ nodes from the path $R = (k_1, \ldots, k_\rho)$. Path p can use at most q of the arcs in any valid outfork inequality defined on path R.

The lemma is illustrated in Figure 1. In this example the outfork inequality is defined for the path $R = (k_1, k_2, k_3, k_4)$ and the sets $S, T_1, \ldots, T_4$. The path $p = (v_1, k_2, k_3, v_2, v_3)$ (where $v_2 \in T_3$) visits two nodes from R and it uses the arcs (k_2, k_3) and (k_3, v_2) from the outfork inequality.

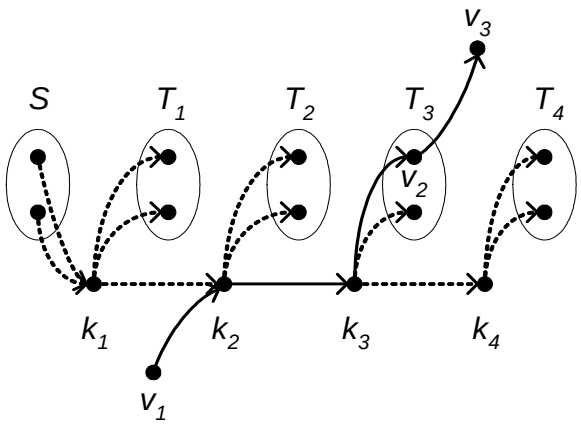

Figure 1: Outfork inequality for $R = (k_1, k_2, k_3, k_4)$ and the path $p = (v_1, k_2, k_3, v_2, v_3)$. Arcs in the outfork inequality are dashed while arcs in path p are solid. The arcs (k_2, k_3) and (k_3, v_2) are used in both the outfork inequality and in path p

Proof. We break path p into subpaths $p_1, \ldots, p_j$ by traversing p from its start and creating a new subpath p_w once a node from R is visited. We then continue to add nodes to p_w as long as p visits nodes from R. When p visits a node outside R, subpath p_w is ended and we start a new subpath p_{w+1} the next time p visits a node from R. Let $|p_w|$ be the number of nodes in subpath p_w ($|p_w| = 1$ is possible). To illustrate the concept of subpaths, consider Figure 2. In this example paths p and R induce the subpaths $p_1 = (k_3, k_4, k_6)$ and $p_2 = (k_1, k_2)$.

It is clear that only the arcs used in the subpaths $p_1, \ldots, p_j$ along with the arcs that path p uses before entering or after leaving path $p_w, w \in \{1, \ldots, j\}$ can be present in the outfork inequality. Consider a subpath p_w that starts in node $k_i, i \neq 1$. The arc in p that is used to enter path p_w cannot be part of the outfork inequality (because the arcs entering k_i in the outfork inequality originate from a node in R). Hence, p_w along with the arcs used to enter and leave p_w can contribute at most $|p_w|$ arcs from the outfork inequality as the path

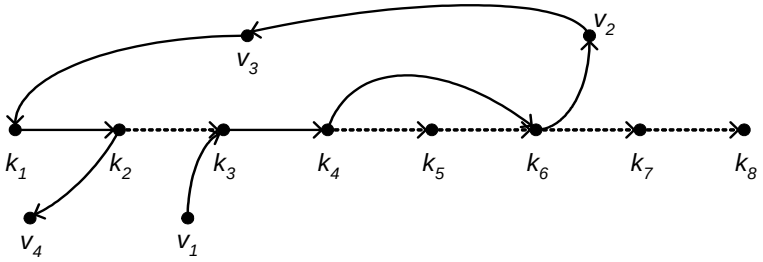

Figure 2: The paths $p = (v_1, k_3, k_4, k_6, v_2, v_3, k_1, k_2, v_4)$ (solid arcs) and $R = (k_1, \ldots, k_8)$ (dashed arcs).

contains $|p_w| - 1$ arcs and the arc used to leave the last node in p_w can be part of the outfork inequality. If a subpath p_w starts in node k_1 then the arc in p that is used to enter path p_w can be part of the outfork inequality. In that event we have to consider two cases: if $p_w = \left(k_1, \ldots, k_{|p_w|}\right)$ then the arc used to leave p_w cannot be part of the outfork inequality as this would make p_w infeasible. If p_w skips some of the first $|p_w|$ nodes from R then the arc used to leave p_w can be part of the outfork inequality but the first arc in p_w used to skip a node in R cannot be part of the outfork inequality as p is a feasible path. Consider for example $p_w = (k_1, k_2, k_5, k_6)$. This subpath skips node k_3 and k_4. If the arc used to enter p_w originates in S then (k_2, k_5) cannot be part of the outfork inequality as the original path p is feasible.

We see that p_w along with the arcs used to enter and leave p_w contributes at most $|p_w|$ arcs, for all $w \in \{1, \ldots, j\}$. Finally, we may conclude that a path that visits q nodes from path R can use at most q arcs from the outfork inequality defined on R. □

Proposition 3. If a vector y^* satisfies the inequalities (14) and (16) where Ω is defined by SP1 then the implied two-index solution x^* satisfies the outfork inequalities (27) defined in Section 4.2.

Proof. Consider a vector y^* that satisfies the conditions in Proposition 3 and assume it violates an outfork inequality defined by a path $R = (k_1, \ldots, k_\rho)$ and sets $S, T_1, \ldots, T_\rho$. Let Φ_k^* be the set of paths from the solution y^* that visit exactly k nodes from the path R. From (2) and (3) we have that $\sum_{i \in R} x^*(\delta^+(i)) = \rho$. Expressing this equality in the y-variables, using the transformation defined in Section 2.3, we get that

$$\sum_{r \in \Phi_1^*} y_r^* + \sum_{r \in \Phi_2^*} 2y_r^* + \ldots + \sum_{r \in \Phi_\rho^*} \rho y_r^* = \rho. \tag{34}$$

From Lemma 2 we know that a path $p \in \Phi_k^*$ can use at most k arcs from the outfork inequality. Thus a path $p_k \in \Phi_k^*$ contributes at most $k y_{p_k}^*$ to the left-hand-side of equation (27). The total contribution from all paths visiting nodes in R is at most

$$\sum_{r \in \Phi_1^*} y_r^* + \sum_{r \in \Phi_2^*} 2y_r^* + \ldots + \sum_{r \in \Phi_\rho^*} \rho y_r^*,$$

which is equal to ρ according to (34). Consequently, the fork inequality cannot be violated. □

The set partitioning formulation using SP1 also implies all infork inequalities (28). The proof is similar to that of Proposition 3 and Lemma 2.

It is clear that the infeasible path inequality (25) is dominated by the fork inequalities and it is therefore implied by the set partitioning formulation using SP1. This infeasible path inequality is, however, not implied by SP2. As an example consider the path $(0, 1, n+1, 2, n+2, 1, n+1, 2, n+2, 2n+1)$. This is a feasible SP2 path and it can be used with value 0.5 in a feasible solution. In such a solution the infeasible path inequality $x_{n+1,2} + x_{2,n+2} + x_{n+2,1} \leq 2$ is violated as the left hand side is equal to 2.5. This also shows that the fork inequality is not implied by SP2.

The strengthened infeasible path inequality (26) is not implied by either formulation (small counter-examples can be constructed easily) and neither are the rounded capacity inequalities or the 2-path inequalities. The SP1 formulation does imply the reachability inequality as shown by the Proposition 4 below. We use the notation

$$\tau(i) = \begin{cases} i & : \ i \in P \\ i - n & : \ i \in D \end{cases}$$

to denote the request corresponding to a node $i \in P \cup D$.

Proposition 4. If a vector y^* satisfies the inequalities (14) and (16) where Ω is defined by SP1 then the implied two-index solution x^* satisfies the reachability inequalities defined in Section 4.5.

Proof. Assume that x^* violates the reachability inequality (31). This implies that there exists a set of conflicting nodes T and a set $S \subseteq P \cup D, T \subseteq S$ such that $\sum_{(j,k)\in(\delta^-(S)\cap A_T^-)} x^*_{jk} < |T|$. Consider a node $i \in T$ and a path p from $\Omega^*_{\tau(i)}$. The subpath p' of p that starts at node 0 and ends at node i uses only arcs from $A_i^- \subseteq A_T^-$ and it crosses $\delta^-(S)$ at least once as $0 \notin S$ and $i \in S$. Consequently every path corresponding to a column $r \in \Omega^*_{\tau(i)}$ contributes at least y^*_r to $\sum_{(j,k)\in(\delta^-(S)\cap A_T^-)} x^*_{jk}$. Because the paths are generated by SP1 we have that $\sum_{r\in\Omega^*_{\tau(i)}} y^*_r = 1$. This implies that that the paths serving i in solution y^* contribute at least 1 to $\sum_{(j,k)\in(\delta^-(S)\cap A_T^-)} x^*_{jk}$. As no path can serve two or more nodes in T we have $\Omega^*_i \cap \Omega^*_q = \emptyset$ for $i, q \in T, i \neq q$ and therefore $\sum_{(j,k)\in(\delta^-(S)\cap A_T^-)} x^*_{jk} \geq |T|$ which is a contradiction. The proof for inequality (32) is similar. □

A solution to the LPM using SP2 can violate reachability inequalities. The example used when discussing the relationship between SP2 and infeasible path inequalities also shows that reachability inequalities can be violated.

Proposition 5. If a vector y^* satisfies the inequalities (14) and (16) where Ω is defined by SP1 or SP2 then the implied two-index solution x^* satisfies the precedence inequalities defined in Section 4.6.

Proof. Assume that x^* violates a precedence inequality. This implies that there exists a request $i \in P$ and a set $S \subset N$ such that $i, 2n+1 \in S$, $0, n+i \notin S$ and $\sum_{(j,k)\in\delta^+(S)} x^*_{jk} < 1$. Every path in Ω^*_i uses at least one arc from $\delta^+(S)$ for every visit to node i as the path

has to visit node $n+i$ before reaching node $2n+1$ or visiting node i again in case of SP2. Therefore, every column $r \in \Omega_i^*$ contributes at least $a_{ir}y_r^*$ to $\sum_{(j,k)\in\delta^+(S)} x_{jk}^*$. Because of equation (14) we have that $\sum_{r\in\Omega_i^*} a_{ir}y_r^* = 1$ and consequently, $\sum_{(j,k)\in\delta^+(S)} x_{jk}^* \geq 1$ which is a contradiction. □

Proposition 6. If a vector y^* satisfies the inequalities (14) and (16) where Ω is defined by SP1 or SP2 then the implied two-index solution x^* satisfies the strengthened precedence inequalities defined in Section 4.7.

Proof. The proof follows that of proposition 5. One should note that an SP1 or SP2 path that visits nodes $i \in P$ and $n+i$ only uses arcs from the set A_i between nodes i and $n+i$ and therefore each column $r \in \Omega_i^*$ contributes at least $a_{ir}y_r^*$ to $\sum_{(j,k)\in(\delta^+(S)\cap A_i)} x_{jk}^*$. □

6 Branch-and-Cut-and-Price Algorithm

In this section, we first describe heuristics for solving the pricing problem. This is followed by separation procedures for the identification of violated valid inequalities in Section 6.2 and by the branching strategy used to explore the enumeration tree in Section 6.3. Implementation issues related to cut and column pool management, and the choice between generating cuts or variables are described in Ropke and Cordeau (2008). Time windows are tightened using the rules described by Dumas et al. (1991) and Cordeau (2006) as well as the window reduction rules for the VRPTW described in Desrochers et al. (1992).

6.1 Pricing problem heuristics

It is well known that the running time of branch-and-price algorithms can be improved by using heuristic algorithms for the pricing problem. As long as the heuristic algorithms are able to find columns with negative reduced cost one can add those columns to the LPM and solve the problem again. Ideally it should be necessary to call the exact pricing algorithm only once for each node in the branch-and-bound tree to verify that no negative reduced cost column exists. In fact, this is not even necessary if the relaxation value associated with a node is lower than the current upper bound. In this case, the lower bound will not be used to fathom the node and it is not necessary to find the optimal relaxation value for this node.

We use several pricing heuristics based on construction algorithms, large neighborhood search (Shaw (1998)) and truncated label setting algorithms. The heuristics and their impact on the branch-and-cut-and-price algorithm are described in Ropke (2005) and Ropke and Cordeau (2008).

6.2 Separation heuristics

In our branch-and-cut-and-price algorithm we use the following five families of inequalities: infeasible path inequalities (26), fork inequalities, reachability inequalities, rounded capacity inequalities and 2-path inequalities. Precedence and strengthened precedence inequalities are not used as they were shown to be implied by the set partitioning formulation in Section 5. We refer the reader to Ropke et al. (2007) for a description of the separation procedures

for the infeasible path, fork and reachability inequalities. Below we describe how the new form of the rounded capacity inequalities as well as the 2-path inequalities are separated.

6.2.1 Separating rounded capacity inequalities

Rounded capacity inequalities are separated by a constructive heuristic called several times with different parameter settings in each call. The heuristic starts with a set S containing only one node (each node is tried as start node several times). Nodes are iteratively added to the set, performing the addition that most increases the objective f described below. The construction is stopped either when a violated inequality is detected or if the current set S is far from violating inequality (29). The objective we seek to maximize with each insertion is the following:

$$\begin{aligned} f(S) &= \lambda_1 \left(\max\left\{q(\pi(S)\setminus S), -q(\sigma(S)\setminus S)\right\} - Qx(\delta^+(S))\right) + \\ &\quad \lambda_2 Q \left(\max\left\{\left\lceil\frac{q(\pi(S)\setminus S)}{Q}\right\rceil, \left\lceil\frac{-q(\sigma(S)\setminus S)}{Q}\right\rceil\right\} - x(\delta^+(S))\right) + \\ &\quad \lambda_3 \left(\min\left\{q(\pi(S)\setminus S), -q(\sigma(S)\setminus S)\right\} - Qx(\delta^+(S))\right), \end{aligned}$$

where λ_1, λ_2 and λ_3 are parameters that control the importance of the three expressions. The reasoning behind the three expressions is the following: when the second expression is larger than 0 then we have discovered a violated inequality; the first expression can be seen as a weakened form of the second that better reflects changes in the demand of the nodes in S; the last expression emphasizes the part of the maximum expression that is not active in the two other expressions (in the hope that further augmentations of S may make this part active). In each call the parameters λ_1, λ_2 and λ_3 are chosen randomly from a uniform distribution on the set $[1,5]\times[1,5]\times[0,1]$, thus giving a higher weight to the two first terms.

6.2.2 Separating 2-path inequalities

The separation heuristic for the 2-path inequalities is a randomized greedy construction heuristic. Starting from a set S containing only one node the heuristic augments S by adding the node that minimizes $x(\delta^+(S))$. In order to randomize the heuristic a random number from the interval $[0, 0.3]$ is added to each x_{ij} when selecting the node to add. For every set S for which $x(\delta^+(S)) < 2$ we check whether a feasible solution can leave S exactly once. To this purpose, we determine whether there exists a feasible path that first visits all the nodes in $\pi(S)\setminus S$, then visits all nodes in S and finally visits all nodes in $\sigma(S)\setminus S$ while satisfying time window, capacity and precedence constraints. This is determined by a straightforward, exact label-setting algorithm. If such a path does not exists then a valid 2-path inequality has been found. The augmentation of S stops when $x(\delta^+(S)) > 3$. All nodes are considered as start nodes and the heuristic is applied several times for each start node. Every time the algorithm has checked if a set S induces a 2-path inequality, the result is stored in a hash-table. This hash-table is examined before checking a set S for feasibility. Thereby we ensure that the same calculation is never performed twice.

6.3 Branching strategy

Branching in a column generation algorithm should be done with care as the branching strategy should preferably be compatible with the algorithm used for solving the pricing problem, i.e., the same type of pricing problem should be solved in the children nodes as in the parent node. This implies that branching decisions should be easily transferred to the subproblem and should not change its structure. In our algorithm we use two branching rules that add a single cut on the x_{ij} variables to the master problem. They are thus compatible with the pricing problems when applying the transformation of the pricing problem described in Section 3.3.

The first branching rule branches on the outflow of a set of nodes as proposed for the CVRP by Naddef and Rinaldi (2002). A set of nodes S is first selected such that $x(\delta^+(S))$ is as far as possible from the nearest integer. Two branches are then created: $x(\delta^+(S)) \leq \lfloor x(\delta^+(S)) \rfloor$ and $x(\delta^+(S)) \geq \lceil x(\delta^+(S)) \rceil$. In our implementation, a good candidate for the set S is found using a simple greedy heuristic, which most often finds a set containing only two nodes.

The second branching rule calculates $x(\delta^+(0))$. If $x(\delta^+(0))$ is fractional then the two branches $x(\delta^+(0)) \leq \lfloor x(\delta^+(0)) \rfloor$ and $x(\delta^+(0)) \geq \lceil x(\delta^+(0)) \rceil$ are created. This rule was proposed by Desrochers et al. (1992) and is often called *branching on the number of vehicles*.

In our implementation we first try to branch on the number of vehicles. If this is not possible because $x(\delta^+(0))$ is integer then we use the first branching rule instead. This choice was motivated by computational experiments documented in Ropke (2005). For both branching rules we always investigate the $\geq$-branch first.

In our branch-and-cut-and-price algorithm, the enumeration tree is explored in a depth-first fashion. We prefer depth-first compared to a best-bound strategy as it uses less memory and each node in the branch-and-bound tree is evaluated slightly faster as we have to perform less work in order to restore a valid basis and to populate the model with useful rows and columns when moving to a new node in the branch-and-bound tree. Furthermore, the depth-first strategy is the easiest to implement. The most significant drawback of the depth-first strategy is that it may visit more nodes than the best-bound strategy if the initial upper bound is poor. Our upper bounds are found using the heuristic described in Ropke and Pisinger (2006) and are usually quite tight (See Section 7 and Table 2).

7 Computational Experiments

This section summarizes the computational experiments performed to assess the performance of our branch-and-cut-and-price algorithm. The algorithm was implemented in C++ and run on an AMD Opteron 250 computer (2.4 GHz) running Linux. CPLEX 9.0 was used as LP solver and the COIN-OR Open Solver Interface (`www.coin-or.org`) was used as an interface to the LP solver. In all experiments, a limit of two hours of CPU time was used unless otherwise indicated.

The algorithm was tested on two sets of instances: instances similar to those introduced by Ropke et al. (2007) (data-set one) and the instances proposed by Li and Lim (2001) (data-set two). The instances proposed by Li and Lim originate from the Solomon VRPTW

instances (Solomon, 1987). For these instances we minimize the total traveled distance in our objective. The Li and Lim instances are divided into two series. Those in the first series have a short planning horizon while those in the second have a longer horizon allowing many requests to be served in the same route. We only report computational results for the first series as only a small subset of the second one could be solved. The pricing problems occurring in the second series are very difficult and the branch-and-cut-and-price algorithm often times out before even obtaining a lower bound at the root node. The integrality gap, on the other hand, was small for the instances that could be solved. See Ropke (2005) for some results on these instances.

The instances introduced by Ropke et al. were produced with a generator similar to that initially proposed by Savelsbergh and Sol (1998). As explained by Ropke et al. (2007), the generator was modified to obtain harder instances by reducing the ratio between the travel times and the length of the planning horizon. In addition, the new generator considers a single depot located at the middle of a square instead of a different depot for each vehicle. In all instances, the coordinates of each pickup and delivery location are chosen randomly according to a uniform distribution over the $[0, 50] \times [0, 50]$ square. The load q_i of request i is selected randomly from the interval $[5, Q]$, where Q is the vehicle capacity. A planning horizon of length $T = 600$ is considered and each time window has width W. The time windows for request i are constructed by first randomly selecting a_i in the interval $[0, T - t_{i,n+i}]$ and then setting $b_i = a_i + W$, $a_{n+i} = a_i + t_{i,n+i}$ and $b_{n+i} = a_{n+i} + W$. In all instances, the primary objective consists of minimizing the number of vehicles, and a fixed cost of 10^4 is thus imposed on each outgoing arc from the depot. The instance generator used by Ropke et al. (2007) contained a bug such that a_{n+i} was set to $b_i + t_{i,n+i}$ instead of $a_i + t_{i,n+i}$. This is unfortunate as it implies that cycling cannot occur with the SP2 algorithm on these instances because the pickup and delivery of each request have non-overlapping time windows. Consequently, we have generated new instances with a corrected version of the instance generator. The new instances appear to be more difficult than those considered by Ropke et al. (2007).

Four groups of instances were generated by considering different values of Q and W. The characteristics of these groups are summarized in Table 1. As in Ropke et al. (2007) we considered ten instances with $30 \leq n \leq 75$ for each group. The name of each instance (e.g., AA50) indicates the class to which it belongs and the number of requests it contains. We repeat the first letter in each instance to distinguish these instances from the ones studied by Ropke et al. (2007). The maximum travel time between two nodes in these instances is $\sqrt{50^2 + 50^2} \approx 70.7$ and the time windows are therefore relatively wide, especially for the CC and DD instances.

Table 1: Characteristics of the new PDPTW instances

Class	Q	W
AA	15	60
BB	20	60
CC	15	120
DD	20	120

For all experiments we use the same numerical precision as Ropke et al. (2007), i.e., travel times and costs are represented using double precision floating point numbers. Upper bounds are found using the adaptive large neighborhood heuristic proposed by Ropke and Pisinger (2006). Table 2 shows the solutions found by the heuristic. The table is split into two major columns. The first major column shows results for data-set one and the second major column shows results for the second data-set. The first 30 instances in data-set two contain 100 requests while the last 6 instances contain 500 requests. For each instance we provide three numbers:

1. A lower bound (LB) on the optimal solution. If the instance has been solved to optimality we report the optimal solution in bold, otherwise we report the best lower bound obtained at the root node by the SP1 or SP2 relaxation. If neither the SP1 nor the SP2 relaxation obtained a lower bound then this entry is left blank. We do not report the lower bound from the branch-and-cut procedure proposed by Ropke et al. (2007) because it usually is quite poor for these hard instances.

2. The solution value found by the heuristic (Heur). If the value is known to be optimal, it is shown in bold. This value, increased by 0.1, is used as an upper bound when testing the branch-and-cut-and-price algorithm in the following sections. We increase the value by 0.1 to ensure that the branch-and-cut-and-price algorithm finds a solution even when the heuristic value was optimal.

3. The time in seconds (Time) the heuristic spent finding the reported solution. The time spent by the heuristic is not included in the computing times reported in the following sections.

Notice that there often is a quite large integrality gap for the instances from the first data-set that have not been solved to optimality (e.g. AA75). We believe that the main reason for this gap is a poor lower bound rather than a weak upper bound.

Our computational experiments focus on three aspects. First, we wanted to measure the impact of the valid inequalities described in Section 4. Second, we wished to investigate the impact of the two subproblems, SP1 and SP2 described in Section 3, on the lower bound and overall solution time. Third, we wanted to compare the performance of our branch-and-cut-and-price algorithm to the branch-and-cut algorithm of Ropke et al. (2007).

7.1 Impact of valid inequalities

The purpose of this section is to investigate the impact of the valid inequalities on the two set partitioning relaxations.

Table 3 reports results obtained by running the algorithm on the first data-set. The table shows results for the SP1 and SP2 relaxations. The column *No cuts* indicates the value of the lower bound as a percentage of the upper bound. It is computed as $100\underline{z}'/\bar{z}$ where $\underline{z}'$ is the lower bound without adding any cuts and $\bar{z}$ is the upper bound (either the optimal solution or the best known solution if the optimal solution is unknown). The rest of the columns report the amount of gap closed using the different families of inequalities: *IPC* – Infeasible path constraints (26), *CC* – Rounded capacity constraints, *2PC* – 2-path

constraints, *FC* – Fork constraints, *RC* – Reachability constraints, *Full* – all constraints. These values are computed as follows: $100(\underline{z} - \underline{z}')/(\bar{z} - \underline{z}')$ where $\underline{z}$ is the lower bound found using the corresponding cuts. Some entries are left blank for instance AA45 because the relaxation without cuts solved the instance to optimality. Note that in some cases the entry in the column *Full* is worse than the value in one of the preceding columns (e.g., AA65, SP1). This can happen when inequalities are separated by heuristics.

The table shows a difference between the SP1 and SP2 relaxations although the difference is rather small. One also notes that the integrality gap at the root node is quite large for many instances (e.g., BB30). The large gap occurs because of the fixed cost on each vehicle and because the LP relaxation is unable to estimate the number of vehicles correctly, i.e., $x(\delta^+(0))$ is lower in the LP relaxation than in a feasible integer solution. One sees that the capacity and 2-path constraints are able to improve the lower bound somewhat, but none of the inequalities are able to improve the lower bound significantly. Future research should aim at finding valid inequalities that increase $x(\delta^+(0))$ in fractional solutions when possible.

Table 4 shows results obtained by applying valid inequalities to the Li and Lim instances with 100 requests. We only report results for the instances for which the lower bound without valid inequalities could be computed within 2 hours and which had a non-zero integrality gap for both relaxations when using a pure column generation approach. When adding valid inequalities, the computing times sometimes exceeded 2 hours. The blank entries for instance LR1_2_6 indicate that the SP1 lower bound was optimal without adding cuts. We see that the valid inequalities are much more useful for this data-set and the 2-path cuts are especially beneficial.

7.2 Branch-and-cut-and-price experiments

This section compares the two set partitioning relaxations to each other as well as to the branch-and-cut algorithm proposed by Ropke et al. (2007). We also test the limits of the algorithms in terms of the instance size that can be solved to optimality. All valid inequalities presented in Section 4 which are not implied by the relaxations are added dynamically to the model.

The results for the first data-set are shown in Table 5. The first two columns show the instance name and the best known upper bound (the optimal solution value in the case of instances solved to optimality and the heuristic value from Table 2 otherwise). The remaining columns indicate $\underline{z}$ – the root node lower bound after adding cuts, *Time* – the total amount of CPU time used (in seconds), *Nodes* – the number of nodes in the branch-and-bound tree, and *Cuts* – the number of cuts added. If the problem was not solved within the time limit the entry in the *Time* column is left blank. If the lower bound could not be computed within the time limit the entry in the $\underline{z}$ column is left blank as well. The tables show results for the branch-and-cut algorithm (B&C) and the branch-and-cut-and-price algorithms using SP1 and SP2 as pricing problems. The row *Solved* indicates how many instances were solved to optimality within the time limit.

Both SP1 and SP2 clearly outperform the branch-and-cut algorithm. The set partitioning formulation using SP1 and SP2 is much better at approximating the number of vehicles in the LP relaxation. The difference in performance between SP1 and SP2 is relatively small.

Table 6 shows results for the Li and Lim (2001) instances in the first series and with

approximately 100 requests. Li and Lim (2001) also proposed instances with 50 requests, but the first series of these instances do not pose any difficulty for the branch-and-cut-and-price algorithm as all instances can be solved in less than 10 minutes (see Ropke (2005)). The instances with 100 requests are harder and for many instances the algorithm fails to solve the root node because the pricing problem is too difficult.

To test the algorithms on large instances we selected instances with approximately 500 requests from the Li and Lim data-set. We chose the six instances that have the tightest time windows. The results of this experiment are shown in Table 7. Three of the six instances could be solved to optimality. To the best of our knowledge, these are the largest PDPTW instances solved to optimality in the literature.

In summary, the SP1 formulation was able to solve 50 of the 76 instances within the time limit, while the SP2 formulation solved 47 instances. We also considered a variant of SP2 without any valid inequalities. This variant, which solved 46 instances, can be seen as a modern implementation of the algorithm of Dumas et al. (1991), with a much improved LP solver and more advanced pricing heuristics. For the instances that were solved by all three configurations, SP1 used on average 375 seconds, SP2 used 723 seconds and SP2 without cuts used 1019 seconds. The improvement over the classic column generation approach is thus not dramatic, but SP1 still manages to solve 9% more instances than SP2 without cuts. These results also indicate that the stronger pricing problem is more important than the addition of valid inequalities when it comes to solving difficult PDPTW instances. Future work on valid inequalities for the PDPTW could of course change this conclusion.

7.3 Further experiments

The purpose of this section is to shed light on how time is spent within the branch-and-cut-and-price algorithm. For this purpose we selected a small set of diverse instances. This set contains CC45 and DD45 from the first data-set and LR1_2_6, LR1_2_9 and LRC1_2_2 from the second data-set. We solved the root node for each of the five instances with both relaxations. The experiment is summarized in Table 8. The columns should be interpreted as follows: *PP*, pricing problem used; *Total time*, time spent solving the root node. *Time preproc.*, time spent on preprocessing, this includes the time spent tightening time windows and calculating the sets A_i^+ and A_i^- needed for the reachability constraint (see Section 4.5); *Time LP*, time spent solving the linear programming relaxation; *Time PP heur.*, time spent solving the pricing problem by heuristic means; *Time PP exact*, time spent solving the pricing problem by the exact method; *Time sep.*, time spent separating valid inequalities; *#Cuts*, number of violated inequalities detected; *#Cols*, number of columns generated; *#PP heur*, number of calls to the pricing heuristic (this is equivalent to the number of iterations in the column generation method); *#PP exact*, number of calls to the exact pricing procedure; *#Labels*, the number of labels processed in the last call to the exact pricing procedure. Every two lines in the table represent information about a particular instance. The first line is for the SP1 relaxation while the second line is for the SP2 relaxation. The last two lines provide averages. All time measurements are in seconds. Notice that the sum of the partial time consumptions does not add up to the total time consumption reported. This is because time is spent in other parts of the algorithm (e.g. management of column and cut pools) that is not reported in the table.

The reason why the SP2 relaxation spends more time on preprocessing compared to SP1, is that the reachability cuts are turned off for SP1 as they are implied by this relaxation. The reachability cut requires the computation of the sets A_i^+ and A_i^- which can be very time consuming, especially for larger instances like the last three. Running the separation procedure for the reachability constraints can also be time consuming as it involves many maximum flow calculations (see Lysgaard (2006)).

8 Conclusions

This paper has introduced a branch-and-cut-and-price algorithm for the PDPTW. We have proved that the most useful valid inequalities (fork and reachability inequalities) from a recently proposed branch-and-cut algorithm (Ropke et al. (2007)) are implied by the set partitioning relaxation SP1. Those inequalities are not implied by the SP2 relaxation, but despite this, the SP1 and SP2 relaxations provide similar lower bounds and are roughly equally hard to compute (for the instances considered in this paper). Even though both the SP1 and SP2 relaxations have been used in the literature before, this paper is the first to present a computational comparison between the two relaxations for the PDPTW. Comparisons between elementary and non-elementary pricing problems for the CVRP and VRPTW have been conducted, for example, by Salani (2005).

For the instances considered in this paper we recommend using the elementary shortest path problem as pricing problem. The pricing problem allows stronger lower bounds and it does not seem to be much harder to solve compared to its non-elementary counterpart. Furthermore, it seems easier to theoretically analyze a set partitioning relaxation based on elementary shortest paths.

We have shown how adding valid inequalities from the 2-index formulation to the master problem interferes with the pricing algorithm. An approach to modify the cost matrix used in the pricing algorithm was proposed to ensure that a strong dominance criterion could be used in the pricing algorithms while adding valid inequalities to the master problem. The experiments concerning valid inequalities showed that the 2-path cut was the most successful of the valid inequalities tested. More research is needed in order to find new families of valid inequalities that close even more of the gap between the lower and upper bounds.

We conclude that several large scale instances can be solved with the current approach but loosely constrained instances remain challenging.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada under grant 227837-04. This support is gratefully acknowledged. We are also thankful to David Pisinger, Stefan Irnich, and three anonymous referees for their valuable comments.

References

E. Balas, M. Fischetti, and W.R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995.

R. Baldacci, N. Christofides, and A. Migozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming, Ser. A*, 115:351–385, 2008.

C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving integer programs. *Operations Research*, 46:316–329, 1998.

N. Boland, J. Detridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34: 58–68, 2006.

J. Bramel and D. Simchi-Levi. Set-covering-based algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 9, pages 85–108. SIAM, Philadelphia, 2002.

J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586, 2006.

J.-F. Cordeau, G. Laporte, J.-Y. Potvin, and M.W.P. Savelsbergh. Transportation on demand. In C. Barnhart and G. Laporte, editors, *Transportation*, Handbooks in Operations Research and Management Science, pages 429–466. Elsevier, Amsterdam, 2007.

M. Dell'Amico, G. Rihigni, and M. Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40:235–247, 2006.

G. Desaulniers, J. Desrosiers, I. Ioachim, M.M. Solomon, F. Soumis, and D. Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, Norwell, MA, 1998.

G. Desaulniers, J. Desrosiers, A. Erdmann, M.M. Solomon, and F. Soumis. VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 9, pages 225–242. SIAM, Philadelphia, 2002.

G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science*, 42:387–404, 2008.

M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6:301–325, 1986.

Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

I. Dumitrescu, S. Ropke, J.-F. Cordeau, and G. Laporte. The traveling salesman problem with pickup and delivery: Polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, 2008. Forthcoming.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R.F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming, Ser. A*, 106(3):491–511, 2006.

S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors, *Column Generation*, chapter 2, pages 33–65. Springer, 2005.

S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18(3):391–406, 2006.

M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008.

N. Kohl, J. Desrosiers, O. B. G. Madsen, M.M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101 – 116, 1999.

A.N. Letchford and J.J. Salazar González. Projection results for vehicle routing. *Mathematical Programming, Ser. B*, 105:251–274, 2006.

H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. In *The 13th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2001*, Dallas, USA, 2001.

Q. Lu and M. Dessouky. An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science*, 38:503–514, 2004.

J. Lysgaard. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 175(1):210–223, 2006.

D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*, chapter 3, pages 53–84. SIAM, Philadelphia, 2002.

G. Righini and M. Salani. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3):255–273, 2006.

G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

S. Ropke. *Heuristic and exact algorithms for vehicle routing problems*. PhD thesis, Department of Computer Science (DIKU), University of Copenhagen, 2005.

S. Ropke and J.-F. Cordeau. Branch-and-cut-and-price for the pickup and delivery problem with time windows. Technical Report CIRRELT-2008-33, HEC Montréal, 2008.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49(4):258–272, 2007.

K.S. Ruland and E.Y Rodin. The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers and Mathematics with Applications*, 33(12):1–13, 1997.

M. Salani. *Branch-and-Price Algorithms for Vehicle Routing Problems*. PhD thesis, Università di Milano, Italy, 2005.

M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

M.W.P. Savelsbergh and M. Sol. DRIVE: Dynamic routing of independent vehicles. *Operations Research*, 46:474–490, 1998.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP-98 (Fourth International Conference on Principles and Practice of Constraint Programming)*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431, 1998.

M. Sigurd, D. Pisinger, and M. Sig. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38:197–209, 2004.

M. Sol. *Column generation techniques for pickup and delivery problems*. PhD thesis, Technische Universitet Eindhoven, 1994.

M.M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

P. Toth and D. Vigo. *The Vehicle Routing Problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, Philadelphia, 2002.

L. A. Wolsey. *Integer Programming*. Wiley-Interscience series in discrete mathematics. Wiley-Interscience publication, 1998.

H. Xu, Z.-L. Chen, S. Rajagopal, and S. Arunapuram. Solving a practical pickup and delivery problem. *Transportation Science*, 37:347–364, 2003.

Name	$\underline{z}$	Cost	Time	Name	$\underline{z}$	Cost	Time
AA30	**31119.1**	**31119.1**	75	LR1_2_1	**4819.1**	**4819.1**	174
AA35	**31299.8**	**31299.8**	93	LR1_2_2	**4093.1**	**4093.1**	212
AA40	**31515.9**	**31515.9**	62	LR1_2_3	3484.1	3486.8	264
AA45	**31759.8**	**31759.8**	124	LR1_2_4		2830.7	411
AA50	**41775.0**	**41775.0**	159	LR1_2_5	**4221.6**	**4221.6**	182
AA55	**41907.8**	**41907.8**	185	LR1_2_6	**3763.0**	**3763.0**	238
AA60	**42140.7**	**42140.7**	118	LR1_2_7		3112.9	272
AA65	**42250.2**	42252.7	135	LR1_2_8		2645.4	383
AA70	**42452.3**	42455.0	187	LR1_2_9	**3953.5**	**3953.5**	193
AA75	43206.5	52472.7	308	LR1_2_10	3376.2	3389.2	225
BB30	**31086.3**	**31086.3**	76	LC1_2_1	**2704.6**	**2704.6**	183
BB35	**31281.2**	**31281.2**	92	LC1_2_2	**2764.6**	**2764.6**	206
BB40	**31493.4**	**31493.4**	63	LC1_2_3	**2772.2**	**2772.2**	230
BB45	**41555.1**	**41555.1**	127	LC1_2_4		2661.4	341
BB50	**41701.0**	41703.4	160	LC1_2_5	**2702.0**	**2702.0**	203
BB55	**41885.7**	**41885.7**	96	LC1_2_6	**2701.0**	**2701.0**	204
BB60	**62420.1**	**62420.1**	178	LC1_2_7	**2701.0**	**2701.0**	221
BB65	**62639.1**	**62639.1**	202	LC1_2_8	**2689.8**	**2689.8**	221
BB70	**62951.0**	62952.3	236	LC1_2_9	**2724.2**	**2724.2**	230
BB75	62232.6	63130.4	256	LC1_2_10	2734.9	2741.6	260
CC30	**31087.7**	**31087.7**	76	LRC1_2_1	**3606.1**	**3606.1**	190
CC35	**31230.6**	**31230.6**	97	LRC1_2_2	**3292.4**	**3292.4**	208
CC40	**31358.5**	**31358.5**	132	LRC1_2_3		3079.5	271
CC45	**31509.1**	**31509.1**	82	LRC1_2_4		2525.8	431
CC50	**41685.3**	41689.0	168	LRC1_2_5	**3715.8**	**3715.8**	208
CC55	**41836.3**	**41836.3**	196	LRC1_2_6	**3360.9**	**3360.9**	198
CC60	37839.7	42015.5	127	LRC1_2_7	3298.2	3317.7	212
CC65	39480.3	42172.1	145	LRC1_2_8	3025.8	3086.7	220
CC70	42124.5	52201.9	288	LRC1_2_9	2995.5	3058.5	229
CC75	43565.0	52375.6	325	LRC1_2_10		2837.5	248
DD30	**21133.3**	**21133.3**	49	LR1101	**56744.9**	56791.7	2278
DD35	**31210.9**	31224.0	99	LR1105	52401.2	52901.3	2403
DD40	**31352.2**	**31352.2**	136	LC1101	**42488.7**	**42488.7**	1870
DD45	**31483.9**	**31483.9**	132	LC1105	**42477.4**	**42477.4**	1880
DD50	**31600.9**	**31600.9**	105	LRC1101	48198.7	48666.4	2145
DD55	**31743.3**	**31743.3**	124	LRC1105		49287.0	2468
DD60	31466.6	41869.4	247				
DD65	35313.7	42125.7	209				
DD70	36690.6	42220.3	175				
DD75	38762.1	42396.8	201				

Table 2: Results from PDPTW Heuristic

	SP1					SP2						
	No cuts	IPC	CC	2PC	Full	No cuts	IPC	CC	2PC	FC	RC	Full
AA30	84.10	0.00	0.00	0.17	0.17	84.10	0.00	0.00	0.17	0.00	0.00	0.17
AA35	84.22	0.00	1.10	1.43	1.42	84.22	0.00	1.02	1.34	0.00	0.00	1.34
AA40	99.97	0.00	100.00	88.32	100.00	99.97	0.00	91.22	80.72	0.00	0.00	93.35
AA45	100.00					100.00						
AA50	81.61	0.00	0.05	0.20	0.20	81.58	0.03	0.05	0.18	0.13	0.14	0.32
AA55	88.26	0.00	0.04	0.08	0.08	88.25	0.00	0.03	0.12	0.13	0.14	0.21
AA60	92.28	0.00	0.13	0.14	0.14	92.20	0.00	0.07	0.12	0.88	0.88	0.97
AA65	94.71	0.00	0.16	0.27	0.16	94.14	4.86	0.92	5.39	5.63	6.49	6.70
AA70	97.14	0.00	0.44	1.58	1.87	96.94	0.96	0.51	4.76	5.04	5.04	6.44
AA75	82.26	0.00	0.09	0.43	0.46	81.91	0.68	0.79	1.17	0.92	0.92	1.58
BB30	72.79	0.00	0.00	1.71	1.71	72.58	0.00	0.00	1.76	0.06	0.01	1.77
BB35	86.50	0.00	0.00	0.49	0.49	86.32	0.00	0.00	1.59	0.06	1.28	1.77
BB40	96.22	0.00	0.00	0.03	0.03	96.18	0.00	0.00	0.24	0.26	0.28	0.57
BB45	78.82	0.00	0.00	0.02	0.02	78.55	0.00	0.00	0.30	0.00	0.00	0.30
BB50	86.16	0.00	0.00	0.04	0.04	86.12	0.00	0.00	0.00	0.03	0.23	0.26
BB55	94.98	0.00	0.00	0.00	0.00	94.67	0.00	0.00	0.03	0.07	0.00	0.08
BB60	87.99	0.00	0.00	0.00	0.01	87.99	0.00	0.00	0.00	0.00	0.00	0.00
BB65	89.26	0.00	0.00	0.00	0.01	89.26	0.00	0.00	0.00	0.00	0.00	0.01
BB70	97.75	0.00	0.01	0.32	0.34	97.75	0.00	0.01	0.33	0.00	0.00	0.35
BB75	98.58	0.00	0.00	0.30	0.31	98.58	0.00	0.00	0.30	0.00	0.00	0.30
CC30	73.37	0.00	0.00	0.09	0.09	72.54	0.00	0.00	0.43	0.00	0.00	0.44
CC35	77.64	0.00	0.00	0.14	0.14	77.09	0.00	0.00	0.14	0.06	0.00	0.17
CC40	80.65	0.00	0.00	0.07	0.07	80.41	0.00	0.00	0.13	0.00	0.00	0.18
CC45	91.84	0.00	0.00	0.00	0.00	91.41	0.00	0.00	0.00	0.05	0.62	0.72
CC50	81.70	0.00	0.00	0.00	0.00	81.66	0.00	0.00	0.03	0.00	0.00	0.02
CC55	87.07	0.00	0.08	0.14	0.14	87.03	0.00	0.04	0.05	0.05	0.00	0.10
CC60	90.06	0.00	0.02	0.01	0.02	89.89	0.00	0.00	0.00	0.00	0.00	0.00
CC65	93.62	0.00	0.00	0.00	0.00	93.38	0.00	0.26	0.29	0.00	0.24	0.40
CC70	80.68	0.00	0.08	0.01	0.08	80.50	0.00	0.04	0.05	0.00	0.03	0.09
CC75	83.17	0.00	0.03	0.03	0.03	82.96	0.00	0.05	0.05	0.00	0.00	0.05
DD30	88.50	0.00	0.03	99.78	100.00	87.92	0.00	0.00	99.57	0.00	0.00	99.85
DD35	68.97	0.00	0.00	0.98	0.98	68.43	0.00	0.00	1.28	0.17	0.07	1.38
DD40	73.80	0.00	0.00	0.18	0.18	73.40	0.00	0.00	0.06	0.06	0.06	0.18
DD45	79.00	0.00	0.13	0.10	0.13	78.75	0.00	0.05	0.13	0.15	0.15	0.29
DD50	84.13	0.00	0.09	0.11	0.13	83.83	0.00	0.48	0.03	0.00	0.00	0.29
DD55	90.83	0.00	0.07	0.10	0.18	90.66	0.00	0.11	0.15	0.00	0.00	0.17
DD60	75.13	0.00	0.03	0.08	0.08	74.94	0.00	0.11	0.11	0.01	0.00	0.12
DD65	83.83	0.00	0.01	0.01	0.01	83.50	0.00	0.00	0.02	0.23	0.22	0.23
DD70	86.90	0.00	0.00	0.00	0.00	86.58	0.35	0.00	0.75	1.15	1.30	1.34
DD75	91.42	0.00	0.00	0.07	0.07	91.04	0.00	0.01	0.11	0.66	0.86	1.00
Avg.	86.40	0.00	2.63	5.06	5.38	86.18	0.18	2.46	5.18	0.41	0.49	5.73

Table 3: Impact of valid inequalities on the first data-set.

	SP1					SP2						
	No cuts	IPC	CC	2PC	Full	No cuts	IPC	CC	2PC	FC	RC	Full
LR1_2_5	99.98	100.00	0.00	0.00	100.00	99.98	100.00	0.00	0.00	0.00	0.00	100.00
LR1_2_6	100.00					99.98	68.85	0.00	100.00	100.00	0.00	100.00
LR1_2_9	99.34	2.45	0.00	72.65	72.81	99.34	2.45	0.00	68.97	0.00	0.00	71.81
LR1_2_10	99.61	0.00	0.00	2.80	2.80	99.43	0.00	0.00	8.83	2.50	2.50	10.35
LC1_2_9	99.63	0.00	0.00	14.15	14.03	99.54	0.00	0.00	2.92	0.00	0.00	2.92
LC1_2_10	99.73	0.00	0.00	10.10	10.10	99.61	0.00	11.07	37.41	3.76	19.03	38.41
LRC1_2_1	99.92	13.27	0.00	100.00	100.00	99.92	13.27	0.00	100.00	0.00	0.00	100.00
LRC1_2_2	99.14	0.00	0.00	80.03	80.03	99.14	0.00	0.00	80.03	0.00	0.00	80.03
LRC1_2_5	99.84	14.04	0.00	16.36	16.36	99.61	12.81	0.00	49.76	3.18	6.15	56.66
LRC1_2_7	99.31	0.00	0.00	14.22	14.57	98.88	0.00	6.32	8.66	7.38	15.05	22.30
LRC1_2_8	97.98	0.00	0.00	15.16	14.94	97.65	0.00	0.00	15.85	0.12	0.04	15.90
LRC1_2_9	97.51	0.00	0.00	17.03	17.03	96.97	0.36	0.48	15.28	0.72	0.97	16.05
Avg.	99.33	10.81	0.00	28.54	36.89	99.17	16.48	1.49	40.64	9.80	3.64	51.20

Table 4: Impact of valid inequalities on the second data-set (instances proposed by Li and Lim (2001)).

		B&C		SP1				SP2			
Name	$\bar{z}$	$\underline{z}$	Time	$\underline{z}$	Time	Nodes	Cuts	$\underline{z}$	Time	Nodes	Cuts
AA30	31119.1	24513.3	6.0	26179.4	6.5	3	7	26179.4	11.2	3	25
AA35	31299.8	26337.5	9.1	26431.6	15.9	5	16	26427.5	34.4	5	25
AA40	31515.9	31501.5	19.3	31515.9	13.4	1	9	31515.2	47.1	5	28
AA45	31759.8	31701.0	1484.4	31759.8	15.9	1	0	31759.8	25.5	1	0
AA50	41775.0	31843.4	832.1	34108.5	72.0	7	11	34105.5	139.6	5	38
AA55	41907.8	33064.2	65.7	36992.7	65.6	3	11	36992.3	125.8	3	23
AA60	42140.7	33351.2	234.2	38892.1	235.7	5	6	38887.5	256.5	5	30
AA65	42250.2	32547.2	519.7	40020.9	349.5	9	5	39940.9	599.7	13	36
AA70	42452.3	32587.7		41262.0	2477.5	75	34	41237.5		160	78
AA75	52472.7	32599.4		43206.5		188	47	43131.4		97	85
BB30	31086.3	21129.0	63.4	22772.9	6.0	3	5	22712.0	9.0	3	8
BB35	31281.2	23659.7	13.9	27078.4	18.0	5	9	27076.2	33.2	7	16
BB40	31493.4	23782.1	32.6	30304.1	18.9	3	6	30298.5	35.8	3	17
BB45	41555.1	23891.8		32754.8	46.3	9	4	32669.7	58.2	7	9
BB50	41701.0	29072.1	482.6	35930.7	192.6	25	14	35929.6	159.9	11	16
BB55	41885.7	27863.2		39781.5	55.8	3	0	39656.8	85.5	3	3
BB60	62420.1	42605.8		54925.9	181.7	27	7	54925.5	276.7	35	6
BB65	62639.1	42901.3		55910.9	294.8	25	4	55910.9	486.2	33	6
BB70	62951.0	44114.8		61537.7	1839.4	139	15	61537.8	2182.8	117	33
BB75	63127.5	44491.3		62232.6		265	34	62232.5		200	45
CC30	31087.7	11134.5		22817.1	11.3	5	1	22588.7	19.2	5	8
CC35	31230.6	11277.9		24257.8	59.1	17	13	24087.3	80.9	11	18
CC40	31358.5	11373.3		25293.8	254.8	29	15	25224.8	658.7	39	74
CC45	31509.1	11514.1		28939.1	175.9	11	5	28822.2	843.8	27	58
CC50	41685.3	11690.2		34058.3	1962.3	137	16	34041.7	4005.2	145	98
CC55	41836.3	11812.0		36432.5	2729.2	117	15	36414.1		180	75
CC60	42015.5	11976.2		37839.7		162	22	37767.7		113	51
CC65	42172.1	12094.7		39480.3		85	21	39391.5		59	33
CC70	52201.9	12198.3		42124.5		44	12	42029.2		36	37
CC75	52375.6	12340.2		43565.0		47	7	43456.6		26	34
DD30	21133.3	11117.6		21133.3	25.2	1	25	21129.5	75.5	5	25
DD35	31210.9	11212.6		21620.2	765.1	143	38	21494.5	2266.8	261	128
DD40	31352.2	11324.1		23152.6	160.8	15	11	23028.4	687.5	37	47
DD45	31483.9	11433.9		24880.6	525.6	31	29	24811.6	1313.3	41	64
DD50	31600.9	11525.2		26593.7	1976.0	77	47	26506.4	4238.8	115	84
DD55	31743.3	11600.6		28836.5	1178.5	25	10	28782.0	3951.6	63	39
DD60	41869.4	11724.0		31466.6		88	10	31389.2		66	44
DD65	42125.7	12018.9		35313.7		59	22	35191.3		42	60
DD70	42220.3	12094.3		36690.6		55	10	36630.7		31	61
DD75	42396.8	12245.1		38762.1		42	13	38635.8		24	65
	#Solved		12		30				28		

Table 5: Branch-and-cut-and-price results on the first data-set.

		B&C		SP1				SP2			
Name	$\bar{z}$	$\underline{z}$	Time	$\underline{z}$	Time	Nodes	Cuts	$\underline{z}$	Time	Nodes	Cuts
LR1_2_1	4819.1	4819.1	287.5	4819.1	5.1	1	0	4819.1	112.4	1	0
LR1_2_2	4093.1	4067.4		4093.1	84.0	1	0	4093.1	214.1	1	0
LR1_2_3	3486.9	3312.0						3484.1		2	1
LR1_2_4	2830.7	2452.8									
LR1_2_5	4221.6	4215.0	1075.5	4221.6	11.7	1	1	4221.6	130.6	1	1
LR1_2_6	3763.0	3482.1		3763.0	1041.6	1	0	3763.0	2198.9	1	2
LR1_2_7	3112.9	2773.2									
LR1_2_8	2645.5	2149.3									
LR1_2_9	3953.5	3815.1		3946.4	418.5	25	29	3946.1	2503.9	93	153
LR1_2_10	3389.2	2879.5		3376.2		4	3				
LC1_2_1	2704.6	2704.6	180.3	2704.6	4.9	1	0	2704.6	116.5	1	0
LC1_2_2	2764.6	2753.8	4979.5	2764.6	27.6	1	0	2764.6	140.2	1	0
LC1_2_3	2772.2	2561.2		2772.2	250.1	1	0	2772.2	240.6	1	0
LC1_2_4	2661.4	1944.6									
LC1_2_5	2702.0	2702.0	223.9	2702.0	6.3	1	0	2702.0	89.5	1	0
LC1_2_6	2701.0	2701.0	580.9	2701.0	9.8	1	0	2701.0	96.1	1	0
LC1_2_7	2701.0	2701.0	423.0	2701.0	10.9	1	0	2701.0	100.1	1	0
LC1_2_8	2689.8	2316.8		2689.8	31.5	1	0	2689.8	118.6	1	0
LC1_2_9	2724.2	1966.8		2715.6	6628.6	91	11	2712.1		73	18
LC1_2_10	2741.6	1493.7		2734.9		9	5	2734.9		9	52
LRC1_2_1	3606.1	3569.1	2485.5	3606.1	12.6	1	3	3606.1	154.1	1	3
LRC1_2_2	3292.4	3026.6		3286.8	1440.4	3	23	3286.8	1053.3	3	21
LRC1_2_3	3079.5	2529.8									
LRC1_2_4	2525.8	2103.1									
LRC1_2_5	3715.8	3333.6		3710.9	517.8	17	7	3709.6	1419.8	27	99
LRC1_2_6	3360.9	3072.7		3360.9	27.7	1	0	3360.9	142.4	1	2
LRC1_2_7	3317.7	2720.4		3298.2		81	25	3289.0		40	629
LRC1_2_8	3086.7	2441.6						3025.8		2	277
LRC1_2_9	3058.6	2261.3		2995.5		2	21	2980.7		2	50
LRC1_2_10	2837.5	2074.2									
	Avg.		1279.5		619.3				552.0		
	Solved	8		17				16			

Table 6: Branch-and-cut-and-price results on the second data-set (instances proposed by Li and Lim (2001)). 100 requests.

		SP1				SP2			
name	UB	$\underline{z}$	time	nodes	Cuts	$\underline{z}$	time	nodes	Cuts
LR1101	56744.9	56740.9	1682.3	3	0	56740.9	2514.8	3	0
LR1105	52901.3	52401.2		8	34	52399.9		8	49
LC1101	42488.7	42488.7	778.9	1	0	42488.7	704.8	1	0
LC1105	42477.4	42477.4	762.7	1	0	42477.4	940.9	1	0
LRC1101	48666.5	48198.7		9	59	48192.7		7	89
LRC1105	49287.1								
	solved	3				3			

Table 7: Branch-and-cut-and-price results on the second data-set (instances proposed by Li and Lim (2001)). Instances with 500 requests and tight time windows.

Name	PP	Time total	Time preproc.	Time LP	Time PP heur.	Time PP exact	Time sep.	#Cuts	#Cols	#PP heur	#PP exact	#Labels
CC45	SP1	33.9	0.1	0.9	30.6	0.8	1.0	0	2306	140	4	90061
	SP2	96.8	6.4	1.3	85.9	0.4	1.9	13	2763	197	5	55492
DD45	SP1	41.8	0.1	1.0	37.3	1.3	1.4	4	2084	183	6	100036
	SP2	88.9	6.2	1.0	78.7	0.7	1.4	11	2192	199	5	84536
LR1_2_6	SP1	1085.0	0.3	0.5	75.4	996.6	5.4	0	1650	127	3	2905131
	SP2	2226.8	90.2	0.4	80.9	2021.9	26.8	2	1679	119	3	4595936
LR1_2_9	SP1	77.4	0.3	0.4	50.0	3.4	15.9	17	1553	142	1	199984
	SP2	367.1	55.3	0.4	66.7	5.6	231.2	21	1431	150	2	214272
LRC1_2_2	SP1	169.1	0.2	0.3	97.7	45.9	17.5	23	1591	152	2	729012
	SP2	681.1	77.1	0.4	151.4	64.8	377.6	21	1620	197	2	922721
Avg.	SP1	281.4	0.2	0.6	58.2	209.6	8.2	8.8	1837	149	3.2	804845
	SP2	692.1	47.0	0.7	92.7	418.7	127.8	13.6	1937	172	3.4	1174591

Table 8: Detailed results for the root node on a subset of instances.